Efficient offloading mechanism for UAVs-based Value Added Services

Sihem Ouahouah*‖ , Tarik Taleb‖ , JaeSeung Song**, and Chafika Benzaid§
* High national School of Computer Science, ESI, Algiers, Algeria. Email: s_ouahouah@esi.dz
‖ Aalto University, Espoo, Finland. Email : tarik.taleb@aalto.fi
** Sejong University, Seoul, South Korea. Email : jssong@sejong.ac.kr
§ LSI, USTHB, Algiers, Algeria. Email : cbenzaid@usthb.dz

***Abstract*—Unmanned Aerial Vehicles (UAVs) are expected to be used everywhere to provision different services and applications, impacting different aspects of our daily lives. Basically, UAVs are characterized by their high mobility. Some may remain motionless for a specific time to perform pre-programmed missions. Whilst UAVs would be used for specific applications, they could additionally offer numerous IoT (Internet of Things) value-added services (VAS) when they are equipped with suitable IoT devices. Many IoT VAS applications require high amount of resources and/or diverse IoT devices that cannot be offered by a single UAV. In order to overcome this limitation, this paper aims to explore, i) the diversity of IoT devices on-board UAVs, and ii) the mobility of UAVs for offering UAVs-based IoT VAS. Two solutions are proposed for carrying out different IoT VAS. Both solutions are modeled using linear integer programming. While the first solution aims to reduce the energy consumption, the second one aims to shorten the response time. The simulation results demonstrate the efficiency of both solutions in achieving their design goals.**

I. Introduction

Unmanned Aerial Vehicles (UAVs), also known as drones, have recently gained a lot of interest from both academia and industry due to their potential in providing different applications from height [1], [2]. Indeed, UAVs can be used in many areas, including smart cities, post mail delivery, transport management, safety monitoring, and disaster managements [1]. UAVs can communicate directly with one another (i.e., in a Device-to-Device fashion) and with a system orchestrator (SO; command and control station) via different wireless technologies, such as WiFi and Long Term Evolution (LTE). As described in [1], SO provides management functionalities for UAVs, including routing, communication, and task assignment. Usually, UAVs are characterized by high mobility, the possibility to carry different Internet of Things (IoT) devices (e.g., sensors and camera) on-board, and to be remotely controlled from the ground by SO [3], [4]. By adding extra IoT devices on-board UAVs, UAVs could deliver numerous value added services (VAS) in the arena of IoT, serving diverse IoT applications from height [5]. However, IoT VAS are resource-greedy and require diverse IoT devices that may not be on board a single UAV. Therefore, a single IoT service may require the involvement of multiple UAVs. Meanwhile, the distribution of the computation of an IoT task among several UAVs helps to extend the individual lifetime of UAVs. To this end, an IoT service assigned to a given UAV could be broken down into a set of sub-tasks where a part of them are offloaded to other UAVs flying in the proximity of the original UAV.

The concept of computation offloading has been widely studied in the literature [6], [7]. Authors in [8] present a computation offloading framework for Android devices. The framework allows mobile devices to decide which parts of an application to be executed locally and which parts to be offloaded to a remote cloud. The benefit of the proposed offloading mechanism was illustrated in terms of application responsiveness. Authors in [9] formulate the offloading decision making problem as a multi-user computation offloading game and devise a distributed algorithm for offloading computation from end-user mobiles to remote mobile-edge cloud in a wireless multi-channel environment. The proposed solution is shown to achieve a Nash equilibrium with high reliability and efficiency. Authors in [10] investigated the inter-vehicles computation offloading issue. An offloading framework that reduces the task completion time is proposed. To select the appropriate vehicles for offloading, four strategies were considered, namely: i) random selection strategy; ii) computing capacity-based selection strategy; iii) distance-based selection strategy; and iv) multi-attributed selection strategy. The simulation results showed that the multi-attributed strategy offered the shortest task completion time.

In this paper, we propose the use of offloading mechanism for handling IoT tasks in a UAV-based IoT platform as envisioned in [11], [12]. This will lighten the computation overhead on UAVs and will provide the opportunity to execute many IoT tasks that may require many IoT devices. To the best of our knowledge, the computation offloading among UAVs has not been studied in the literature yet. The main contribution of this work is the design of two solutions to efficiently handle different IoT services using computation offloading mechanisms. In the proposed solutions, each IoT task would be partitioned into a set of sub-tasks that can be executed simultaneously among a cluster of UAVs. Each IoT task would be assigned to the cluster head that, in turn, will execute a set of sub-tasks and offload the remainder sub-tasks to its cluster members. The sub-tasks will be assigned to UAVs based on their power supply, resources in terms of memory and CPU computation, and their on-board IoT devices. Both solutions are modeled using linear integer programming [13]. While the first solution aims to reduce the energy consumption

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

under the constraint of executing IoT tasks within a tolerable delay, the second solution seeks to reduce, as much as possible, the response time under the energy constraint. The simulation results demonstrate the feasibility and ability of both solutions in achieving their design goals.

The rest of this paper is organized as follows. In Section II, we formally define the problem of increasing the UAVs lifetime and reducing the response time for UAVs-based IoT. The proposed solutions are presented in Section III. The simulation results are presented in Section IV. Finally, we conclude the paper in Section V.

II. System model and problem formulation

Fig. 1 shows a general overview of the envisioned UAV-based IoT platform. It mainly consists of three essential components: i) the UAVs and their on-board IoT devices; ii) the SO that is in charge of remotely managing different UAVs and their on-board IoT devices; and iii) the transport network that ensures the communication among UAVs, as well as the communication between UAVs and the SO [3], [4]. Basically, SO is in charge of i) managing the routes for all UAVs, ii) selecting adequate UAVs to accomplish different IoT tasks, and iii) deciding on how to forward and process data from different UAVs. The transport network can be established through any wireless technology, such as $3G$ and $4G$. While UAVs would be used for specific missions (e.g., parcel delivery), they could offer simultaneously numerous IoT VAS when they are equipped with suitable IoT devices [5]. Using UAVs to deliver IoT VAS would lower both capital and operational expenses in the near future, and that is for provisioning new IoT services that can be offered from height [3].

In this paper, we devise two solutions that aim to efficiently manage the computation load of IoT applications among multiple UAVs. These solutions aim to minimize the response time of UAVs and save their energy consumption without interrupting the original mission of each UAV. Let Γ denotes the set of UAVs in the network. Each UAV $u \in \Gamma$ is equipped with a set of IoT devices that will be involved in carrying out an IoT service. Let $\mathcal{S}(u)$ denotes the set of IoT devices of $u \in \Gamma$. Let us assume that an IoT service consists of a set of tasks $\Omega = \bigcup_{i=1\cdots\mathcal{N}} T_i$, where T_i denotes the ith task of an IoT service. We also assume that if required, each task T_i can be subdivided into a set of sub-tasks $\mathcal{P}(T_i)$ that can be executed simultaneously. Formally, $T_i = \bigcup_{T_{i,j} \in \mathcal{P}(T_i)} T_{ij}$. If two sub-tasks cannot be scheduled simultaneously, we simply consider them as one sub-task. Each task T_i requires a set of IoT devices $\mathcal{S}(T_i)$, and each sub-task $T_{i,j}$ requires a set of IoT devices $\mathcal{S}(T_{i,j})$. Formally, $\mathcal{S}(T_i) = \bigcup_{T_{i,j} \in \mathcal{P}(T_i)} \mathcal{S}(T_{i,j})$. Moreover, we assume that each task T_i and each sub-task $T_{i,j}$ require, respectively, a specific amount of resources $\mathcal{R}(T_i)$ and $\mathcal{R}(T_{i,j})$ in terms of memory and CPU (i.e., computation power). For the sake of simplicity and without loss of generality, we assume that $\mathcal{R}(T_i) = \sum_{T_{i,j} \in \mathcal{P}(T_i)} \mathcal{R}(T_{i,j})$.

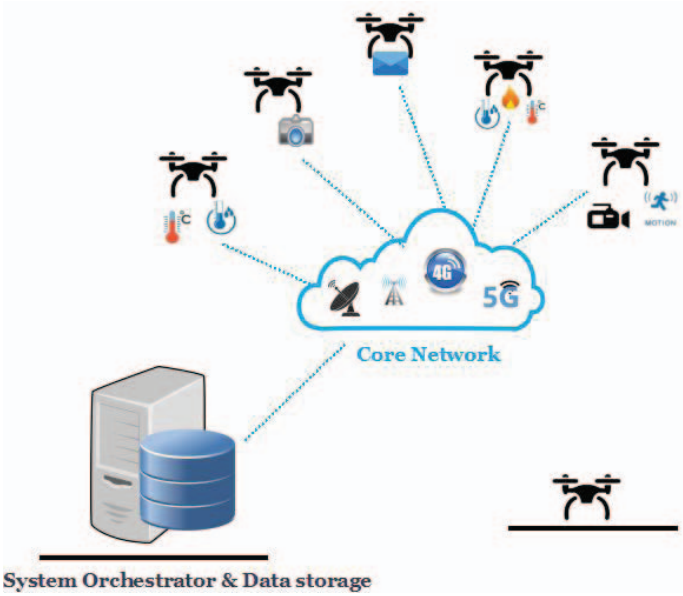


Fig. 1. The envisioned UAV-based IoT platform (similar to [1]).

Basically, UAVs are characterized by their mobility. They can be programmed to execute certain missions (e.g., mail delivery). While they are flying close to each other, they can be also instructed to carry out some extra IoT tasks. We refer to this time when they can reach each other as the *link expiration time*.

For this purpose, we model the network of UAVs in which UAV pairs can communicate among themselves during the link expiration time as a weighted graph $G(V, E, \omega)$. V denotes the vertex set (nodes) of the graph G which refers to the set of UAVs in the network, whereas E denotes the edges set of the graph G which refers to the communication links between pairs of UAVs. An edge $(u, v) \in E$ indicates that nodes u and v can communicate for a specific period denoted by $\omega_{u,v} \in \omega$, which refers to link expiration time between u and v.Note that $\omega_{u,v}$ is computed by taking into account the communication time between UAVs and the SO. The notations used throughout this paper are summarized in Table I.

III. Efficient energy consumption and response time for computation offloading among UAVs

A. Solutions overview

In what follows, Algorithm 1 is used to explain the functionality of the proposed solutions. As described in the previous section, we assume that our system contains a set of UAVs flying under the control of SO. Hence, Algorithm 1 is executed by SO to handle the different IoT tasks. Algorithm 1 takes as input the set of active UAVs Γ, the set of IoT tasks Ω and the link expiration time (among the target set of UAVs) presented as a weighted graph $G(V, E, \omega)$. The output of Algorithm 1 is the set of clusters where the different IoT tasks should be executed. Fig. 2 serves as an example to

TABLE I
NOTATIONS USED IN THIS PAPER.

Notation	Description
T_i	Task i.
$T_{i,j}$	The sub-task j of T_i.
$\mathcal{P}(T_i)$	Set of sub-tasks of Task T_i.
$\mathcal{S}(T_{i,j})$	The devices required to execute $T_{i,j}$.
$\mathcal{S}(T_i)$	The devices required to execute T_i. Formally, $\mathcal{S}(T_i) = \bigcup_{\forall T_{i,j} \in \mathcal{P}(T_i)} S(T_{i,j})$
$R(T_i)$	The resources required to execute T_i.
$R(T_{i,j})$	The resources required to execute $T_{i,j}$.
$n(u)$	Function that returns the neighbors of UAV u.
$\eta(u)$	The set of u's neighbors including also u. Formally, $\eta(u) = n(u) \cup \{u\}$
$\omega_{u,v}$	Link expiration time between UAVs u and v.
$\xi(T_{i,j})$	Energy needed to execute the sub-task $T_{i,j}$.
$\beta(u)$	The amount of energy in UAV u.
$C(u)$	The amount of computation capacity in UAV u.
$X_{i,j}[u,v]$	A boolean variable that is set to 1 if UAV u selects v as cluster head for handling IoT VAS sub-task $T_{i,j}$, 0 otherwise.
$Y_i[u]$	A boolean variable that is set to 1 if u is the cluster head in charge for IoT task i, 0 otherwise.

illustrate the operations of both solutions. In the figure, the link expiration time between nodes (i.e., UAVs) is depicted as weighted dashed edges in $G(V, E, \omega)$. A dashed arrow indicates the directed edge in the weighted directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W})$. The solid arrow between a pair of UAVs u and v indicates that u is a member in a cluster headed by v.

Let $\mathcal{CH}s$ denotes the set of clusters which is initially set to empty (Algorithm 1: Line 1). First, a weighted directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W})$ is generated from graph $G(V, E, \omega)$ by replacing each edge $(u, v) \in E$ by two arrows (u, v) and $(v, u) \in \mathcal{E}$ (Algorithm 1: Lines $2 - 7$). Figs 2 (a) and 2 (b) illustrate how $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W})$ is constructed from $G(V, E, \omega)$. Then, for each IoT task ($T_i \in \Omega$), a cluster $\mathcal{CH}$ is formed to handle it (Algorithm 1: Lines $8 - 12$). More details on $createCluster(T_i, \mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W}))$ which defines the cluster in charge of IoT task T_i, will be given in the next sub-section.

For instance, in Fig. 2 (c), nodes $\{1, 2, 5, 9\}$ form the first cluster with node 9 being the cluster head. Similarly, nodes $\{3, 4, 5\}$ in Fig. 2 (d) constitute the second cluster with node 4 being the cluster head. Let the first cluster handle the IoT task T_1 that is constituted of three sub-tasks $T_{1,1}$, $T_{1,2}$ and $T_{1,3}$ whose execution times are, respectively, $3s$, $2s$ and $1s$. The sub-task $T_{1,1}$ is assigned to UAV 1 because the link expiration time between UAV 1 and its cluster head 9 is $8s$, which is long enough to perform the sub-task $T_{1,1}$ and send back the results to the cluster head. Likewise, sub-tasks $T_{1,2}$ and $T_{1,3}$ are attributed to UAVs 2 and 5, respectively.

Before moving to the next IoT task, the weighted directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W})$ is updated to reflect how much each UAV is busy with the previous tasks (Algorithm 1: Line 10). To this end, the weights of outgoing edges from nodes involved in the execution of sub-tasks are decreased by the duration of their sub-tasks. As depicted in Fig. 2 (d), once the first cluster is formed, the weights of outgoing edges from UAVs 1, 2 and 5 are decreased by 3, 2 and 1, respectively. When the weight of a directed edge $(u, v) \in \mathcal{E}$ becomes 0, the edge is removed

Algorithm 1 Efficent offloading mechanism for UAVs-based VAS

Input:
Γ: A set of UAVs in the network.
Ω: A set of tasks.
$G(V, E, \omega)$: a weighted graph that represents the network topology. V is the set of UAVs in the network, E is the set of links between different UAVs, and ω, the weight of G, is the link expiration time between UAVs. Formally, $V = \Gamma$.
Output:
$\mathcal{CH}s$: Set of clusters that will handle different tasks of Ω.

1: $\mathcal{CH}s = \emptyset$;
//Construct the directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W})$
2: $\mathcal{V} = V$;
3: $\mathcal{E} = E$;
4: **for all** $(u, v) \in E$ **do**
5: $\quad \mathcal{W}_{u,v} = \omega_{u,v}$;
6: $\quad \mathcal{W}_{v,u} = \omega_{u,v}$;
7: **end for**
8: **for all** $T_i \in \Omega$ **do**
9: $\quad \mathcal{CH} = createCluster(T_i, \mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W}))$;
10: $\quad updateGraph(G(V_1, E_1, \omega_1))$;
11: $\quad \mathcal{CH}s = \mathcal{CH}s \cup \{\mathcal{CH}\}$;
12: **end for**
13: **return** $\mathcal{CH}s$;

from $\mathcal{G}$. This means that UAV u cannot process additional sub-tasks before the expiration of the communication link (u, v). Thus, u cannot select v as cluster head for upcoming tasks. Fig. 2 (d) shows that the edge $(1, 6)$ is removed from $\mathcal{G}$ after the formation of the first cluster. Following the same approach, the second cluster is formed in the second iteration of the loop (Algorithm 1: Lines $8 - 12$). As depicted in Fig. 2 (d), the second cluster is formed to handle the second IoT task T_2 that is constituted of two sub-tasks $T_{2,1}$ and $T_{2,2}$ whose execution times are, respectively, $2s$ and $3s$. The sub-tasks $T_{2,1}$ and $T_{2,2}$ are attributed to UAVs 5 and 3, respectively. Note that a UAV can be a member and/or cluster head in many IoT tasks. As shown in Fig. 2 (e), UAV 5 is a member of both clusters. When all clusters are formed, SO sends the network information and the action plan to all concerned UAVs. Upon receiving the results of sub-tasks, the cluster head sends the final results to SO.

B. Solutions description

As aforementioned, two solutions are proposed for inter-UAVs computation offloading. According to the observation that UAVs' energy can be already exhausted by their original missions, the first solution (*ETO-VAS*: Energy aware optimal task offloading) aims to extend the UAVs' lifetime by saving the remaining energy. For this purpose, *ETO-VAS* is designed to be as much lightweight as possible, to avoid assuming extra overheads. The second solution (*DTO-VAS*: Delay aware optimal task offloading) aims to reduce the response time, which is suitable for delay-sensitive IoT applications. Both *ETO-VAS* and *DTO-VAS* are the implementations of the $createCluster(T_i, \mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{W}))$ function. The choice of solution to implement depends on the end-user requirements.

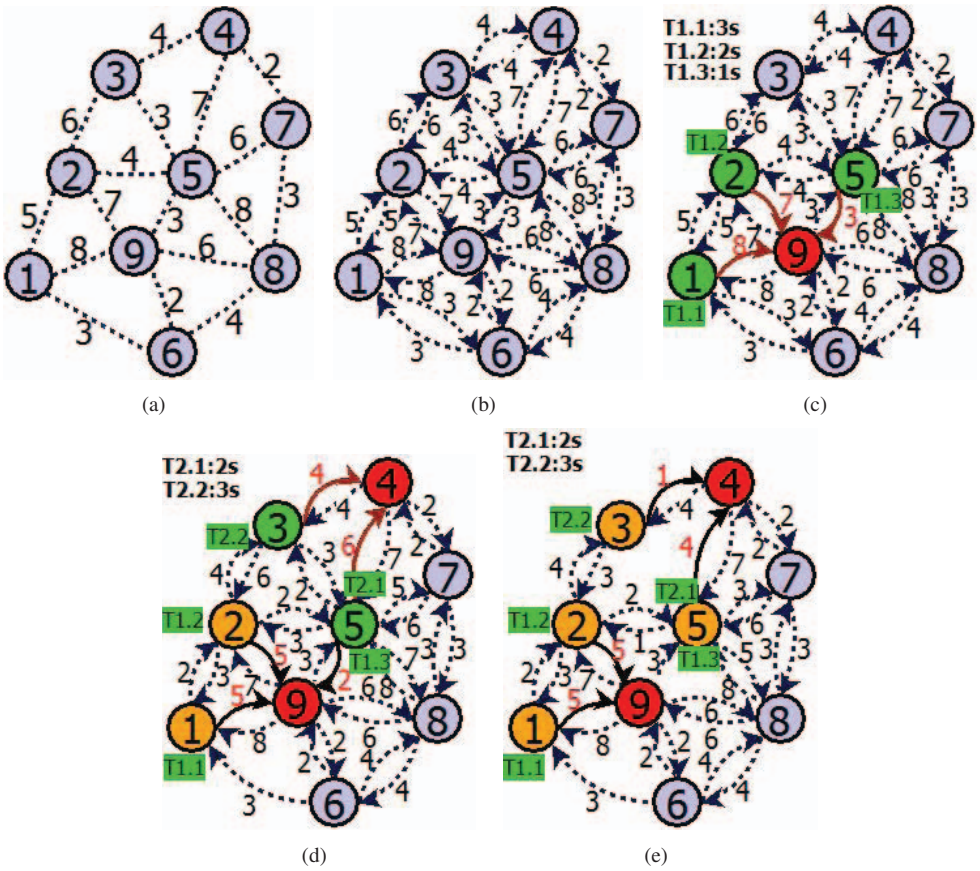


Fig. 2. Illustrative example that shows the execution steps of the proposed solution.

In both solutions, UAVs are organized into clusters; each of them handling an IoT task $T_i \in \Omega$. The cluster head, in charge of an IoT task T_i, offloads sub-tasks of T_i to its cluster members. Note that the cluster head can also execute locally some sub-tasks. Once the sub-tasks are carried out, their results are sent back to the cluster head. After receiving all sub-results, the cluster head aggregate the results and send the final result of task T_i to SO.

1) Energy-aware optimal task offloading: In this sub-section, *ETO-VAS* is described. Recall that *ETO-VAS* aims to increase, as much as possible, the UAVs lifetime. To achieve this, energy-aware task offloading-problem is formulated as a *max-min* integer linear programming problem. As mentioned in Table I, two variables are defined, namely: *i)* $Y_i[v]$, which is a decision boolean variable indicating whether a UAV $v \in \Gamma$ is the cluster head for Task $T_i \in \Omega$. Formally, $Y_i[v]$ is set to 1 if v is the cluster head of T_i. *ii)* $X_{i,j}[u,v]$, which is a decision boolean variable that is set to 1 if cluster head v offloads the sub-task $T_{i,j} \in \mathcal{P}(T_i)$ to cluster member u. *ETO-VAS* is modeled as follows:

$$\max \min_{\forall u \in \Gamma} \beta(u) - \sum_{T_{i,j} \in \mathcal{P}(T_i)} \xi(T_{i,j}) \cdot \sum_{v \in \eta(u)} X_{i,j}[u,v] \quad (1)$$

Subject to

$$\sum_{u \in \Gamma} Y_i[u] = 1 \quad (2)$$

$$\forall u \in \Gamma, \forall v \in \eta(u), \forall T_{i,j} \in \mathcal{P}(T_i) : X_{i,j}[u,v] \leq Y_i[v] \quad (3)$$

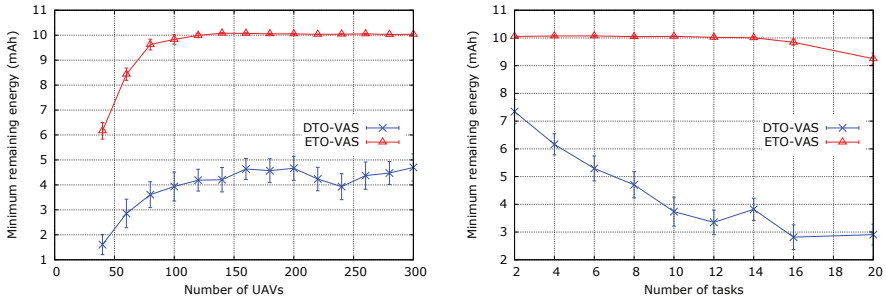
$$\forall u \in \Gamma, \forall v \in \eta(u), \sum_{\forall T_{i,j} \in \mathcal{P}(T_i)} \frac{R(T_{i,j})}{C(u)} \cdot X_{i,j}[u,v] \leq \omega_{u,v} \quad (4)$$

$$\forall T_{i,j} \in \mathcal{P}(T_i) : \sum_{u \in \Gamma} \sum_{v \in \eta(u)} X_{i,j}[u,v] = 1 \quad (5)$$

$$\forall T_{i,j} \in \mathcal{P}(T_i), \forall s \in \mathcal{S}(T_{i,j}) : \sum_{\forall u \in \Gamma, s \in \mathcal{S}(u), \forall v \in \eta(u)} X_{i,j}[u,v] = 1 \quad (6)$$

$$\forall u \in \Gamma : \sum_{\forall T_{i,j} \in \mathcal{P}(T_i)} \sum_{\forall v \in \eta(u)} \frac{R(T_{i,j})}{C(u)} \cdot X_{i,j}[u,v] \leq \mathcal{D} \quad (7)$$

The objective function (1) aims at maximizing, as much as possible, the UAVs' lifetime. This will be ensured by maximizing the minimum remaining energy of UAVs. The first constraint (2) ensures that every IoT task should be handled within only one cluster, and consequently guarantees that there is only one cluster head for each task T_i. The second constraint (3) ensures that the flow generated by sub-task $T_{i,j}$ should be forwarded from the cluster member u to the cluster head v. Whereas, the third constraint (4) ensures that a cluster member u has enough capacity to handle the sub-task $T_{i,j}$ and communicate the results to the cluster head v before the link expiration time $\omega_{u,v}$ is elapsed. Equation (5) enforces



(a) Number of UAVs. (b) Number of tasks.

Fig. 3. Minimum remaining energy at UAVs.

that each sub-task $T_{i,j}$ is executed by only one member. The constraint, defined by Equation (6), ensures that the member selected to handle the sub-task $T_{i,j}$ has the required IoT devices. Constraint (7) ensures that the IoT task T_i is executed within a tolerable delay $\mathcal{D}$. Note that UAV v is selected from $\eta(u) = n(u) \cup \{u\}$, where $n(u)$ is the set of u's neighbors. This will allow the cluster head to execute some sub-tasks locally.

2) Delay-aware optimal task offloading: In this sub-section, *DTO-VAS* is described. Recall that *DTO-VAS* seeks to reduce, as much as possible, the task response time. Similarly to the previous solution, *DTO-VAS* is formulated through an integer linear programming, and its model follows the *min-max* approach. The *DTO-VAS* formulation is as follows:

$$\min \max_{\forall u \in \Gamma} \sum_{\forall T_{i,j} \in \mathcal{P}(T_i)} \sum_{\forall v \in \eta(u)} \frac{R(T_{i,j})}{C(u)} \cdot X_{i,j}[u,v] \tag{8}$$

Subject to

$$\sum_{u \in \Gamma} Y_i[u] = 1 \tag{9}$$

$$\forall u \in \Gamma, \forall v \in \eta(u), \forall T_{i,j} \in \mathcal{P}(T_i) : X_{i,j}[u,v] \leq Y_i[v] \tag{10}$$

$$\forall u \in \Gamma, \forall v \in \eta(u), \sum_{\forall T_{i,j} \in \mathcal{P}(T_i)} \frac{R(T_{i,j})}{C(u)} \cdot X_{i,j}[u,v] \leq \omega_{u,v} \tag{11}$$

$$\forall T_{i,j} \in \mathcal{P}(T_i) : \sum_{u \in \Gamma} \sum_{v \in \eta(u)} X_{i,j}[u,v] = 1 \tag{12}$$

$$\forall T_{i,j} \in \mathcal{P}(T_i), \forall s \in \mathcal{S}(T_{i,j}) : \sum_{\forall u \in \Gamma, s \in \mathcal{S}(u), \forall v \in \eta(u)} X_{i,j}[u,v] = 1 \tag{13}$$

$$\forall u \in \Gamma : \beta(u) - \sum_{T_{i,j} \in \mathcal{P}(T_i)} \xi(T_{i,j}) \cdot \sum_{v \in \eta(u)} X_{i,j}[u,v] \geq \mathcal{E}_\epsilon \tag{14}$$

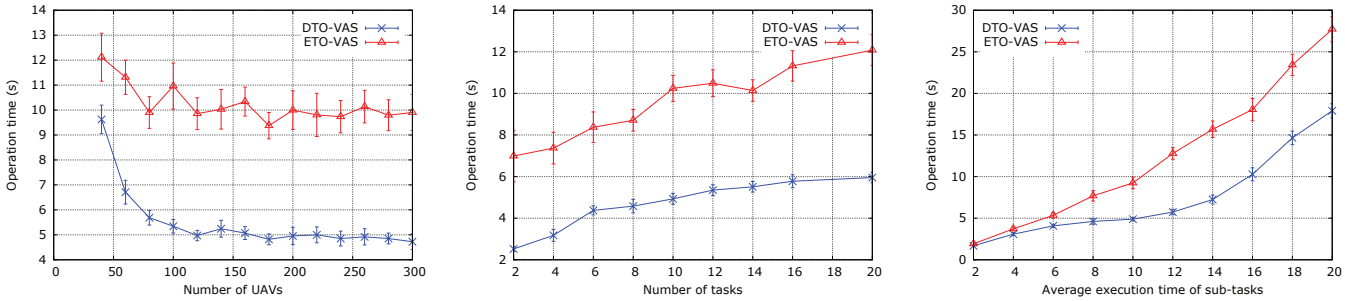
The objective (8) function aims at minimizing the maximum task execution delay at each UAV. Constraints (9) $\cdots$ (13) are similar to those in the previous solution. Constraint (14) ensures that the remaining energy in each UAV u is higher than a predefined threshold $\mathcal{E}_\epsilon$ in order to extend the UAVs lifetime when reducing the response time.

IV. Simulation results

To the best of the authors' knowledge, the computation offloading among UAVs has not been studied in the literature. For this reason, in this study, only the performances of *DTO-VAS* and *ETO-VAS* are compared. Both solutions are implemented through Gurobi optimizer [14] and their performances are assessed through simulation. The algorithms are evaluated in terms of the following metrics: i) the minimum remaining energy of UAVs after the execution of each solution; ii) the operation time (i.e., response time) that is defined as the maximum time required for executing all IoT tasks. The obtained results are depicted in Figs. 3 and 4, where each point in the plots is the average of 40 independent runs. Three types of experiments are conducted: i) the number of UAVs is varied while both the number of tasks and the average execution time of a sub-task are set to 10; ii) the number of tasks is varied while the number of UAVs and the average execution time of each sub-task are, respectively, set to 150 and 10; iii) the average execution time of a sub-task is varied while setting the number of UAVs to 150 and the number of tasks to 10.

A. Minimum remaining energy of UAVs

Fig. 3 shows the impact of the number of UAVs and the number of tasks on UAVs lifetime. Fig. 3 (a) plots the minimum remaining energy as a function of the number of UAVs. The results demonstrate that an increased number of UAVs has a positive impact on UAVs' lieftime in both solutions. This can be explained by the fact that increasing the number of UAVs will increase the probability of selecting UAVs with higher power supply. Moreover, Fig. 3 (a) clearly shows that ETO-VAS outperforms *DTO-VAS* in terms of minimum remaining energy of UAVs. Indeed, *ETO-VAS* extends the UAVs lifetime by more than 100% compared to *DTO-VAS*. Fig. 3 (b) shows that the number of IoT tasks has a negative impact on the UAVs lifetime. The more the IoT tasks are, the higher the energy consumption will be. It is also observed from Fig. 3 (b) that unlike *DTO-VAS*, the UAVs lifetime is decreased smoothly for *ETO-VAS* . In fact, *ETO-VAS* extends the UAVs lifetime by more than 300% compared to *DTO-VAS* when the number of UAVS is more than 16.



(a) Number of UAVs. (b) Number of tasks. (c) Average duration of sub-tasks.

Fig. 4. Operation time for executing tasks.

B. Operation time for executing tasks

Fig. 4 depicts the simulation results of the operation time for both solutions. The obtained results show the superiority of *DTO-VAS* over *ETO-VAS* with regard to the maximum time required for executing all IoT tasks. This is due to the fact that *DTO-VAS* is designed to reduce the operation time. From Fig. 4 (a), it is observed that the number of UAVs has a positive impact on the operation time. The more the UAVs in the network are, the more likely UAVs with higher computational capacity will be selected. Fig. 4 (a) shows that *DTO-VAS* reduces the operation time by more than 100% compared to *ETO-VAS*. Meanwhile, Figs. 4 (b) and 4 (c) show that increasing the number of IoT tasks and the average duration of sub-tasks leads to a negative impact on the operation time. Indeed, the more the IoT tasks are, the longer the operation time will be. Furthermore, the increase of sub-tasks' duration will necessarily increase the operation time at each UAV. From Figs. 4 (b) and 4 (c), it becomes clear that *DTO-VAS* provides significantly lower operation time compared to *ETO-VAS*.

V. Conclusion

In this paper, we considered the problem of increasing the UAVs lifetime and reducing the response time by leveraging computation offloading among UAVs for carrying out IoT services. Two solutions, named *ETO-VAS* and *DTO-VAS*, have been proposed. Both solutions use a linear integer programming formulation. *ETO-VAS* aims to maximize the UAVs lifetime by electing the UAVs with higher power supply. In this way, more energy will be saved and the UAVs lifetime will be extended. Meanwhile, *DTO-VAS* aims to reduce the response time by favoring the selection of UAVs with more resource capacities. The simulation results have proven the efficiency of each solution in achieving its key design objectives.

Acknowledgement

Prof. Song was supported by the Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No.B0184-15-1001).

References

[1] N. H. Motlagh, T. Taleb, and O. Arouk, "Low-altitude unmanned aerial vehicles-based internet of things services: Comprehensive survey and future perspectives," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 899–922, 2016.

[2] S. Koulali, E. Sabir, T. Taleb, and M. Azizi, "A green strategic activity scheduling for uav networks: A sub-modular game perspective," *In Proc.of 2017 IEEE Communications Magazine*, vol. 54, no. 5, pp. 58–64, May 2016.

[3] N. H. Motlagh, M. Bagaa, and T. Taleb, "Uav-based iot platform: A crowd surveillance use case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, 2017.

[4] H. Motlagh, N, M. Bagaa, T. Taleb, and J. Song, "Connection steering mechanism between mobile networks for reliable uav's iot platform," *In Proc.of 2017 IEEE International Conference on Communications (ICC '17)*, May 2017.

[5] ——, "Uav selection for a uav-based integrative iot platform," *In Proc.of 2016 IEEE Global Communications Conference (GLOBECOM '16)*, pp. 1–6, Dec 2016.

[6] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation of- floading for mobile systems, mobile networks and applications," *ACM J Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, Feb 2013.

[7] A. Fahim, A. Mtibaa, and K. A. Harras, "Making the case for computational offloading in mobile device clouds," *In Proc. of the 19th ACM annual international conference on Mobile computing & networking (MobiCom '13)*, pp. 203–205, 2013.

[8] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: a computation offloading framework for smartphones," *In Proc. of the Second International Conference on Mobile Computing, Applications, and Services (MobiCASE '10)*, vol. 76, pp. 59–79, 2010.

[9] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct 2016.

[10] B. Li, Y. Pei, H. Wu, Z. Liu, and H. Liu, "Computation offloading management for vehicular ad hoc cloud," *In Proc. of the 14th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP '14)*, pp. 728–739, Aug 2014.

[11] A. Benslimane, T. Taleb, and R. Sivaraj, "Dynamic clustering-based adaptive mobile gateway management in integrated vanet-3g heterogeneous wireless networks," *IEEE.Selected Areas in Communications*, vol. 29, no. 3, pp. 559–570, Mar 2011.

[12] T. Taleb and A. Benslimane, "Design guidelines for a network architecture integrating vanet with 3g & beyond networks," *In Proc.of 2010 IEEE Global Communications Conference (GLOBECOM '10)*, pp. 1–5, Dec 2010.

[13] M. Bagaa, M. Younis, and I. Balasingham, "Optimal strategies for data aggregation scheduling in wireless sensor networks," *In Proc. of 2015 IEEE Global Communications Conference (GLOBECOM '15)*, pp. 1–6, Dec 2015.

[14] gurobi. [Online]. Available: http://www.gurobi.com/